Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claims 1-104. (cancelled)

105. (currently amended) A computer system for automatically generating a simulation model ~~of~~ for a selected configuration of software simulation elements, comprising:

storage means for storing a plurality of said software simulation elements, said plurality of software simulation elements provided with inter working connections so as to constitute the simulation model of an architecture, each said software simulation element representing a component~~wherein the simulation model comprises models of integrated circuits for the realization of a machine that conforms to a functional specification of a configuration defined in a configuration definition file~~; and

a data processing system comprising execution means provided with configuration means that comprises:

means for creating a simulation of wiring by executing stored regular expressions,

means for using ~~the~~ a configuration definition file, a component and connection rule table, and a connection coherency rule table[[;]], wherein the component and connection rule table and the connection coherency rule table are written in a high level language (HLL), and the component and connection rule table describes properties of ~~software~~ said components for simulating at

least one of ~~the~~ a plurality of integrated circuits, and

means for instantiating components ~~resulting from~~based on the configuration definition file, ~~and~~

an HLL code generator that combines parameters of the components with connection rules of the component and connection rule table, and

means for automatically generating source code files comprising the simulation model corresponding to the selected configuration specified by the configuration definition file;

wherein the simulation model comprises software simulation elements each corresponding to an integrated circuit which together comprise the design of a processing machine that conforms to a functional specification of the selected configuration as defined in the configuration definition file.

106. (currently amended) ~~A~~The system according to claim 105, wherein the components comprise Active Components, Monitoring and Verification Blocks, Intermediate Blocks, System Blocks, and Global Blocks.

107. (currently amended) ~~A~~The system according to claim 106, further comprising means to perform a conformity check of ~~the~~ connections by comparing an instance connection table with a table of coherency rules for ~~the~~ physical connections between the models chosen ~~from~~ for ~~the~~ blocks constituting the simulation model.

108. (currently amended) ~~A~~The system according to claim 107, further comprising means to compare the instance connection table to the connection coherency rule table to detect any incompatible connections between ~~the~~ ends of ~~the~~

connections between blocks, and in cases where an incompatible connection is detected, the system is configured to specify and add an adapter component (Intermediate Block) to the instance connection table, said adapter component being inserted into the detected incompatible connection between the components.

109. (currently amended) ~~A~~The system according to claim 108, wherein the component and connection rule table includes properties of the components and contains parameters common to all of the component types and exists in the form of a table distributed into one or more associative tables, wherein ~~the~~ table entries are names designating ~~all of the~~ possible models for the same component.

110. (currently amended) ~~A~~The system according to claim 109, wherein the associative tables are adapted to contain a description either in the form of parameter sets or in the form of references to procedures that generate ~~the required~~a set of values, and

~~the~~ entries of ~~these~~ associative tables comprise names each of which ~~designating~~ designates ~~all of~~a the possible model[[s]] for the same component and form a character string containing predetermined special identifiers that are replaced by values calculated by the configuration means.

111. (currently amended) ~~A~~The system according to claim 110, wherein at least three selectors indicate the instance to be used, and in which the following selectors are transmitted as parameters to a constructor of an HLL object:

a first selector indicating a current instance (item);

a second selector specifying the current instance connected to ~~the other~~an end of ~~the~~ a port; and

a third selector indicating a composite instance corresponding to an active Component containing an observation port.

112. (currently amended) ~~A~~The system according to claim 105, wherein the configuration means further comprises:

one or more connection coherency rule tables representing the rules for interconnecting the components and for inserting intermediate components;

one or more component and connection rule tables representing the system-level connection rules and the rules for generating connections between the signals; and

one or more source file formatting tables representing the rules for generating instances of HLL objects.

113. (currently amended) ~~A~~The system according to claim 105, wherein the configuration means further comprises:

an HLL base class uniquely identifying each object instantiated ~~and polling the configuration~~;

means for generating and automatically instantiating System Blocks;

means for using tables to associate the signals connected together under a unique name of the connecting wires; and

means for using a formatting table to generate hardware description language (HDL) and HLL source files.

114. (currently amended) ~~A~~The system according to claim 105, wherein the system is configured to receive from an operator a functional specification of the configuration in a high level language, and to complete the functional specification

with the components in a language ~~having a level lower~~other than said high level language.

115. (currently amended) ~~A~~The system according to claim 105, wherein the following entries in a hash define a Component Type and correlate each Component Type to the hash, said hash comprising the following:

a first entry comprising a name of a hardware description language (HDL) module of a component and a name of a corresponding source file; and

a second entry comprising a definition of a method for selecting ~~the~~ signals that are part of a Port, said definition comprising a set of entries indexed by a name of the Port;

wherein the configuration means is configured to associate each said Port name with a table of regular expressions and a pointer to a signal connection procedure that controls the application of the expressions to the names of ~~the~~ signals of ~~the~~ an interface of the component.

116. (currently amended) ~~A~~The system according to claim 115, wherein said Component Type comprises one or more Active Components having a generic structure that includes a containing Block that contains an HDL Block including an HDL description and a Block in HLL that provides access paths to HDL resources and a description of the containing block in the high level language;

wherein the set of signals of the HDL Block constitutes an interface of the containing Block, formed by one or more Ports~~, which are arbitrary~~ comprising logical selections of ~~the~~ signals of ~~an~~ the interface, and also formed by interface adapters which ~~are the software parts that handle~~provide, in each said Port, ~~the~~ two-

way communication between the ~~parts in~~ high level language and ~~those in~~the hardware description language ~~the~~ for interface adapters being selected by the configuration means.

117. (currently amended) ~~A~~The system according to claim 116, wherein the Ports are specified in the form of regular expressions that select subsets of signals to be connected and define connection rules.

118. (currently amended) ~~A~~The system according to claim 105, wherein the configuration means is further configured to generate Transfer Components which are inserted ~~on~~ to be operable at each side of ~~the~~ an interface between servers, said Transfer Components comprising wires for inputs and registers for outputs.

Claims 119-129 (cancelled).

130. (currently amended) A method for automatically generating a simulation model ~~of~~ for a selected configuration of software simulation elements, comprising:

storing a plurality of said software simulation elements, said plurality of software simulation elements provided with inter working connections so as to constitute the simulation model of an architecture, each said software simulation element representing a component ~~wherein the simulation model comprises models of integrated circuits for the realization of a machine that conforms to a functional specification of a configuration defined in a configuration definition file~~;

creating a simulation of wiring by executing stored regular expressions;

using ~~the~~ a configuration definition file, a component and connection rule table, and a connection coherency rule table, wherein the component and connection

rule table and the connection coherency rule table are written in a high level language, and the component and connection rule table describes properties of ~~software~~ said components for simulating at least one of ~~the~~ a plurality of integrated circuits;

instantiating components ~~resulting from~~ based on the configuration definition file; ~~and~~

combining, via ~~an~~ a high level language (HLL) code generator, the parameters of the components with the connection rules of the component and connection rule table; and

automatically generating source code files comprising the simulation model corresponding to the selected configuration specified by the configuration definition file,

wherein the simulation model comprises software simulation elements each corresponding to an integrated circuit which together comprise the design of a processing machine that conforms to a functional specification of the selected configuration as defined in the configuration definition file.

131. (currently amended) ~~A~~The method according to claim 130, wherein the components comprise Active Components, Monitoring and Verification Blocks, Intermediate Blocks, System Blocks, and Global Blocks.

132. (currently amended) ~~A~~The method according to claim 131, further comprising performing a conformity check of ~~the~~ connections by comparing an instance connection table with a table of coherency rules for ~~the~~ physical connections between ~~the~~ models chosen from the blocks to ~~constituting~~ constitute the simulation model.

133. (currently amended) ~~A~~The method according to claim 132, further comprising:

comparing the instance connection table to the connection coherency rule table to detect any incompatible connections between the ends of the connections between blocks; and

in cases where an incompatible connection is detected, specifying and adding an adapter component (Intermediate Block) to the instance connection table, said adapter component being inserted into the detected incompatible connection between the components.

134. (currently amended) ~~A~~The method according to claim 133, wherein the component and connection rule table includes properties of the components and contains parameters common to all of the component types and exists in the form of a table distributed into one or more associative tables, and ~~the~~ entries being names designating all ~~of the~~ possible models for the same component.

135. (currently amended) ~~A~~The method according to claim 134, wherein the associative tables are adapted to contain a description either in the form of parameter sets or in the form of references to procedures that generate ~~the required~~a set of values, and

wherein ~~the~~ entries of ~~these~~ associative tables comprise names each of which designat~~inges~~ ~~all of the~~a possible model[[s]] for the same component, and form a character string containing predetermined special identifiers that are replaced by calculated values.

136. (currently amended) ~~A~~The method according to claim 135, further comprising:

indicating, using at least three selectors, the instance to be used; and

transmitting the following selectors as parameters to a constructor of an HLL object:

a first selector indicating a current instance (item);

a second selector specifying the current instance connected to ~~the other~~an end of ~~the~~ a port; and

a third selector indicating a composite instance corresponding to an active Component containing an observation port.

137. (currently amended) ~~A~~The method according to claim 130, further comprising:

representing, by one or more connection coherency rule tables, the rules for interconnecting the components and for inserting intermediate components;

representing, by one or more component and connection rule tables, the system-level connection rules and the rules for generating connections between the signals; and

representing, by one or more source file formatting tables, the rules for generating instances of HLL objects.

138. (currently amended) ~~A~~The method according to claim 130, further comprising:

uniquely identifying, via an HLL base class, each object instantiated ~~and polling the configuration~~;

generating and automatically instantiating System Blocks;

using tables to associate the signals connected together under a unique name of the connecting wires; and

using a formatting table to generate the hardware description language and HLL source files.

139. (currently amended) ~~A~~The method according to claim 130, further comprising:

receiving, from an operator, a functional specification of the configuration in a high level language; and

completing the functional specification with the components in a language ~~having a level lower~~ other than said high level language.

140. (currently amended) ~~A~~The method according to claim 130, further comprising:

defining, using the following entries in a hash, a Component Type; and

correlating, using the following entries in the has, each Component Type to the hash, wherein said hash comprises the following:

a first entry comprising a name of a hardware description language (HDL) module of a component and a name of a corresponding source file; and

a second entry comprising a definition of a method for selecting the signals that are part of a Port, said definition comprising a set of entries indexed by a name of the Port; wherein

said method further includes associating each said Port name with a table of regular expressions and a pointer to a signal connection procedure that controls the application of the expressions to the names of the signals of the interface of the component.

141. (currently amended) ~~A~~The method according to claim 140, wherein said Component Type comprises one or more Active Components having a generic structure that includes a containing Block that contains an HDL Block including an HDL description and a Block in HLL that provides access paths to HDL resources and a description of the containing block in the high level language;

wherein the set of signals of the HDL Block constitutes an interface of the containing Block, formed by one or more Ports~~, which are~~comprising arbitrary logical selections of ~~the~~ signals of an interface, and also formed by interface adapters which ~~are the software parts that handle~~provide, in each said Port, ~~the~~ two-way communication between the ~~parts in~~ high level language and ~~those in~~the hardware description language.

142. (currently amended) ~~A~~The method according to claim 141, further comprising specifying the Ports in the form of regular expressions that select subsets of signals to be connected and define connection rules.

143. (currently amended) ~~A~~The method according to claim 130, further comprising generating Transfer Components which are inserted ~~on~~ to be operable at each side of ~~the~~an interface between servers, said Transfer Components comprising wires for inputs and registers for outputs.